Using the Max-Sum Algorithm for Supply Chain Emergence in Dynamic Multiunit Environments

Maria Chli and Michael Winsper

***Abstract*—Supply chain formation (SCF) is the process of determining the set of participants and exchange relationships within a network with the goal of setting up a supply chain that meets some predefined social objective. Many proposed solutions for the SCF problem rely on centralized computation, which presents a single point of failure and can also lead to problems with scalability. Decentralized techniques that aid supply chain emergence offer a more robust and scalable approach by allowing participants to deliberate between themselves about the structure of the optimal supply chain. Current decentralized supply chain emergence mechanisms are only able to deal with simplistic scenarios in which goods are produced and traded in single units only and without taking into account production capacities or input-output ratios other than 1:1. In this paper, we demonstrate the performance of a graphical inference technique, max-sum loopy belief propagation (LBP), in a complex multiunit unit supply chain emergence scenario which models additional constraints such as production capacities and input-to-output ratios. We also provide results demonstrating the performance of LBP in dynamic environments, where the properties and composition of participants are altered as the algorithm is running. Our results suggest that max-sum LBP produces consistently strong solutions on a variety of network structures in a multiunit problem scenario, and that performance tends not to be affected by on-the-fly changes to the properties or composition of participants.**

***Index Terms*—Max-sum algorithm, mechanism design, supply chain formation (SCF).**

I. Introduction

SUPPLY chains are the networks of organizations involved in the conversion of raw materials, information or services into a end product, and the sale of this product to an end consumer. Supply chain formation (SCF) is the process of determining the set of participants in a supply chain, the set of goods which are exchanged between the participants, and the terms of these exchanges [1]. The traditional manual approach to the formation of supply chains involves time-consuming periods of contract tendering and negotiations. As uncertain markets and an increasing need to capitalize on emerging business opportunities combine to necessitate increased speed in commercial decision-making, there exists a growing need for automated support.

Manuscript received September 12, 2013; accepted July 26, 2014. This paper was recommended by Associate Editor M. Jeng.

M. Chli is with the Department of Computer Science, Aston University, Birmingham B4 7ET, U.K. (e-mail: m.chli@aston.ac.uk).

M. Winsper was with the Department of Computer Science, Aston University, Birmingham B4 7ET, U.K. He is now with the Centre for Supply Chain Improvement, University of Derby, Derby DE22 1GB, U.K. (e-mail: m.winsper@derby.ac.uk).

Digital Object Identifier 10.1109/TSMC.2014.2351782

Computational approaches to SCF generally model potential supply chain participants—businesses capable of forming a link in the supply chain—as individual computational agents with limited information about the structure of the supply chain as a whole. These agents express their capabilities and costs through a mechanism, typically negotiations or auctions, determining the subset of agents capable of forming the most efficient supply chain. At the conclusion of this process, which is typically completed in a fraction of the time required of the manual approach, the supply chain is formed.

Agent-based approaches to SCF may either be centralized or decentralized. Centralized approaches typically make use of combinatorial auctions to determine allocations, with the NP-hard winner determination problem (WDP) usually being solved with integer programming. The use of integer programming implies complete knowledge by some party about the bids of all agents; this is an assumption which might not always be practical or allowable in the real world. Centralized approaches also introduce a single point of failure into the process as well as potential problems with scalability. Decentralized approaches to the SCF problem make only minimal assumptions about the participating agents, giving them a wide range of applicability, allowing for greater scalability than centralized approaches, but posing challenges in determining allocations, given that agents only possess local information about the structure of the network and the capabilities of other participants.

In [2], it was shown that when the SCF problem is cast as an optimization of a pairwise cost function, max-sum loopy belief propagation (LBP) is capable of quickly finding consistently optimal allocations in a decentralized manner over a wide range of network structures. However, this paper examines a very simple scenario in which goods are only traded one unit at a time, production capacities are not accounted for, input to output ratio is always assumed to be 1:1 and grants agents little autonomy—agents are forced to actively participate for the entirety of the process and are unable to change any of their properties, such as the price they wish to charge for the goods they sell or their capacity, once the process has begun.

In this paper, we propose a framework for the representation of realistic supply chains in which multiple units of several goods are produced and traded. The max-sum LBP-based technique for decentralized SCF presented in [2] is used as a starting point and it is extended to accommodate the multiunit, variable capacity and input-output ratio scenario. We also present a set of experiments demonstrating the performance

of LBP in a truly dynamic environment, in which we test the performance of the mechanism when faced with changes to the properties (e.g., capacity, reserve price, etc.) and composition of participants (e.g., departure of a participant or entrance of a new participant to the system). Our results demonstrate that the enriched max-sum algorithm is capable of producing optimal or near-optimal allocations over a range of network topologies in both static and dynamic environments.

In Section II, we provide details of previous agent-based techniques for SCF, focusing on their ability to deal with multiunit exchanges, additional constraints such as factory capacities, and dynamic settings. In Section III, we describe our framework for the representation of multiunit supply chains, and in Section IV, we explain how the max-sum LBP algorithm is applied to multiunit SCF. Section V details our experimental procedure and our model of the dynamic environment, while Section VI shows the results produced by LBP in both static and dynamic multiunit environments. In Section VII, we provide conclusions about our work, and suggest some potential directions for future work.

II. Background

Multiagent systems enable us to model a number of properties characteristic of supply chains, including decentralized decision making by self-interested agents and the process of self-organization by participants. It is no surprise, then, that application of the agent-based paradigm to several aspects of supply chains has been an ongoing focus of multiagent systems research for several years, particularly in supply chain management [3], [4], most notably the trading agent competition in supply chain management game [5], and in the area of SCF [6]. The majority of the literature on SCF involving self-interested agents deals with the use of market-based approaches to elicit costs and capabilities from participants, with auctions being the most frequently used of these methods.

In this section, we examine previous auction-based approaches to SCF and introduce the use of probabilistic graphical models and LBP for SCF and related problems.

A. Auctions

Many approaches to SCF involve modeling the supply chain as a network of auctions, with first and second-price sealed bid auctions, double auctions and combinatorial auctions among the most frequently-used methods. SCF through auctions is a popular approach for a number of reasons: auctions are frequently used in real-world tendering and sales situations, they are often able to form good solutions to the SCF problem, and some auctions are able to guarantee various desirable game-theoretic properties. Such properties include incentive compatibility, which means that a participant's dominant strategy is to truthfully reveal its private information; individual rationality, where participants are guaranteed to receive nonnegative utility by participating, budget balance, where payments made to and by the mechanism are equal, and allocative efficiency, where the utility of participants is maximized. It is important to note, however, that it is impossible for a two-sided mechanism to satisfy each of these four properties simultaneously [7]. We examine the game theoretic properties of our approach in Section VI-F.

Perhaps the most comprehensive series of studies on SCF using auctions comes from Walsh *et al*. and Babaioff and Walsh, who examine the efficiency of supply chains formed using simultaneous double auctions [1], combinatorial auctions [9], and one-shot double auctions [8], respectively.

1) Double Auctions: Walsh and Wellman [1] proposed a market protocol with bidding restrictions referred to as simultaneous ascending (M+1)st price with simple bidding (SAMP-SB), which uses a series of simultaneous ascending double auctions. SAMP-SB was shown to be capable of producing highly-valued allocations—solutions which maximize the difference between the costs of participating producers and the values obtained by participating consumers—over several network structures, although it frequently struggled on networks where competitive equilibria did not exist. The authors also proposed a similar protocol, SAMP-SB-D, with the provision for decommitment in order to remedy the inefficiencies caused by solutions in which one or more producers acquire an incomplete set of complementary input goods and are unable to produce their output good, leading to negative utility. This use of a post-allocation decommitment stage was recognized as an imperfect approach, however, due to the possible problems created by rendering the results of auctions as nonbinding.

In [8], a one-shot double auction mechanism, referred to as Trade Reduction auctions, is proposed based upon work in [10], that sacrifices perfect allocative efficiency, in order to guarantee incentive compatibility, individual rationality and budget balance. The authors propose both a centralized and a distributed algorithm for determining allocations; however, their distributed algorithm relies on the use of mediators for each good, communication between these mediators, and a central coordinator agent. These factors combine to indicate an assumption of centralization which may not always be valid.

2) Combinatorial Auctions: Walsh *et al*. [9] use a combinatorial auction protocol on a subset of the networks in [1] to attempt to find allocations under strategic bidding behavior by agents. Combinatorial approaches to SCF hold the advantage of being able to avoid the problem of dead ends in the presence of input complementarities by allowing agents to bid for bundles of goods. Due to the strategic bidding behaviors adopted by the agents in [9], the results of the combinatorial protocol did not represent a significant improvement on the double auction protocol, with the quality of the solutions found to be influenced in large part by the amount of available surplus in the networks.

More recent work examining the multiunit case of SCF has seen the proposal of mixed multiunit combinatorial auctions (MMUCAs) for SCF [11], with the standard combinatorial model of bids being placed for bundles of goods replaced by negotiations over "transformations," essentially commitments by bidders to produce a set of output goods given a set of input goods. There exist several approaches to solving the NP-hard WDP associated with MMUCAs, and the quality of the solutions produced by these techniques tends to depend on the characteristics of the network being tested [12]. Although all existing MMUCA solvers rely on

integer programming and thus may face difficulties with scalability, Giovannucci *et al*. [13] have improved the applicability of MMUCAs to larger SCF problems by proposing an integer program mapping which improves the computational efficiency of the WDP calculation by taking advantage of the structural properties of the network.

In [14], continuing with the framework of combinatorial auctions, auctioneers are equipped with a Petri-net formalism which under certain conditions (e.g., acyclic networks) lead to optimal formations. Finding a local, decentralized solver for MMUCAs and the Petri-net formalism remains an ongoing area of research [11], [14].

3) Suitability of Auctions for Multiunit and Dynamic Environments: Both double and combinatorial auctions are readily generalizable to multiunit environments, although application to this case presents combinatorial auctions in particular with a very hard problem. The most frequently-studied form of combinatorial auctions, including all of the approaches mentioned in Section II-A2, are one-shot mechanisms i.e., there exists no scope for participants to change their properties or to enter or leave—each participant places a single set of bids, and the process of computing a solution to the problem begins immediately. Periodic auction-based approaches, such as the SAMP-SB double auction protocol from [1], permit the departure of existing participants or the entry of new ones during the bidding process, and allow participants to change their bids—although typically with some restrictions.

B. Loopy Belief Propagation

Although auctions and negotiations are by far the most commonly-employed techniques in agent-based approaches to the SCF problem, LBP has been used as a method for task allocation for several years in the related area of agent-based decentralized coordination [15], [16]. In [2], an LBP-based approach was applied to the SCF problem, noting that the passing of messages in LBP is comparable to the placing of bids in standard auction-based approaches. The decentralized and distributed nature of LBP also allows for the avoidance of the scalability issues present in centralized approaches such as combinatorial auctions.

LBP is a decentralized and distributed approximate inference scheme involving the application of Pearl's belief propagation algorithm [17] to graphical models containing cycles. It uses iterative stages of message passing as a means for estimating the marginal probabilities of nodes being in given states: at each iteration, each node in the graph sends a message to each of its neighbors giving an estimation of the sender's beliefs about the likelihoods of the recipient being in each of its possible states. Nodes then update their beliefs about their own states based upon the content of these messages, and the cycle of message passing and belief update continues until the beliefs of each node become stable.

1) Probabilistic Graphical Models: Probabilistic graphical models are a means for encoding probability distributions over a set of variables using graphs [18]. Graphical models may be directed or undirected. Directed graphical models, known as Bayesian networks (BNs), represent qualitative dependence between variables—an arc from node i to node j indicates that i causes j, as well quantitative statistical dependence—an arc between nodes also corresponds to a conditional probability of the state of a child node given its parent(s)—$p(x_j|x_i)$ represents the probability of x_j given x_i. Undirected graphical models—Markov random fields (MRFs)—are useful for representing symmetric dependencies between variables. In MRFs, as in BNs, adjacency (through an undirected edge) of nodes indicates dependence, while nodes which are not directly connected are strictly independent.

2) Max-Sum Loopy Belief Propagation: The most commonly used version of LBP, the sum-product algorithm, is used to estimate marginal probabilities at individual nodes. Because we are interested in finding the optimal state configuration of the network as a whole rather than the most likely state of any one node, we use a well-known variant of LBP, the max-sum algorithm, to estimate the maximum *a posteriori* (MAP) assignment of our supply chain networks. Max-sum LBP is well-suited as a means for allocation determination in SCF for a number of reasons. First, as mentioned earlier, the formalism introduced in [1] for the representation of supply chains as task dependency networks—bipartite directed acyclic graphs with nodes representing producers, consumers and goods linked by edges representing potential flows of goods, allows for easy conversion into pairwise MRFs suitable for inference once explicit representation of goods in the graphs is removed. Replacing the process of bidding in auctions with message passing between agents allows participants to share their beliefs about the optimal structure of the supply chain without revealing any more private cost information than they would in an open auction. LBP operates in a decentralized and distributed manner, properties important for the realistic representation of separate self-interested business entities. Finally, LBP is able to quickly and reliably produce exact results in tree-structured and single-cycle networks while still producing good approximations in more loopy networks.

3) Loopy Belief Propagation Properties: While LBP is known to converge to exact results in a finite number of iterations on tree-structured graphs, there is no such guarantee for more loopy graphs, and if convergence is reached, the solution will be an approximation, unless the graph contains only a single loop [19]. Work by Vinyals *et al*. [20] has established worst-case bounds on the quality of solutions produced by max-sum LBP, although these guarantees hold only when all unary and pairwise potentials are nonnegative, which is not the case in our model. Despite these limitations, our application of LBP to decentralized SCF is shown to outperform alternatives (see Section VI). LBP has seen great success in a number of areas, including turbo codes [21] and low density parity check codes [22], stereo vision [23], as well as in the related field of communication in sensor networks [15], [24]. The application of LBP in supply chain problems is becoming more widespread with work in [25] putting forward binary factor graphs to address scalability concerns associated with larger networks. This effort, however, did not succeed in producing supply chains whose efficiency approximates that of centralized allocations. Further work, Penya-Alba *et al.* [26] use LBP and mediators in a decentralized manner to achieve higher solution quality. All these approaches are limited to single-unit product exchanges.

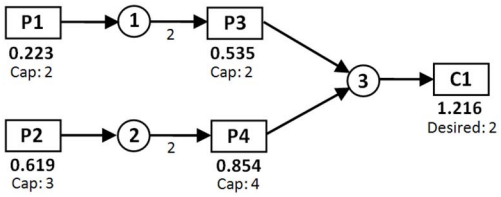

Fig. 1. Simple supply chain TDN from [1] extended to the multiunit case. Producers (P1, P2, P3, P4) and a consumer (C1) are represented by rectangles, while goods are represented by circles. Edges between vertices indicate potential flows of goods. Numbers immediately below producers represent reserve prices, while numbers immediately below consumers indicate consumption values. The values given below reserve prices and consumption values indicate production capacities for producers and desired consumable good quantities for consumers, respectively, and are measured in whole units of the good in question. Edges from goods to producers are labeled with the producer's input to output ratio for that good.

III. Model

The use of task dependency networks (TDNs) for the representation of supply chains was originally proposed in [1]. They offer an effective shorthand for SCF problems and the networks introduced in Section III-D have been used as a benchmark in the literature to facilitate comparison. TDNs allow for a compact representation of the characteristic features of SCF. The first of these features is hierarchical subtask decomposition. Supply chain participants are specialized and are only capable of completing specific tasks (i.e., producing a certain type of good). In order to complete their task, they are often reliant on the completion of subtasks (the production of their input goods) by producers upstream in the supply chain. The second, resource contention, means that multiple participants may rely on common resources, such as goods produced upstream in the supply chain. If these resources are scarce, then these participants may be unable to participate simultaneously in the supply chain. This serves to constrain the number of possible solutions for a given set of participants. Resource scarcity also leads to the exposure problem, a situation in which participants acquire an incomplete set of their input goods and are thus unable to produce their output good.

For the first time, we extend the TDN representation with input to output good ratios, production capacities and consumer desired good quantities in order to model the multiunit case. This requires an example of the extended representation, as shown in Fig. 1. Production capacities and consumer-desired good quantities are measured in whole units of the good in question. A producer with a single input and an input ratio of 2 for that good requires two units of that good in order to produce one unit of its output, four units of that good to produce two of its output, and so on.

In our example, we see that producer P1 is able to produce up to 2 units of good 1, at a cost of 0.223 for each unit it produces. Producer P3 requires 2 units of good 1 (as signified by the edge from good 1 to P3) to produce a single unit of its output good, good 3. Although P3 has the capacity to produce up to 2 units of good 3, this would require P3 to obtain 4 units of good 1, which is not possible in this network instance given P1's maximum output capacity of 2. Similarly, producer P2 is able to produce up to 3 units of good 2 at a cost of 0.619 per unit, and producer P4 requires 2 units of this good in order to produce one unit of good 3. Consumer C1 desires a maximum of 2 units of good 3, and obtains a consumption value of 1.216 for each unit of good 3 that it acquires.

A. Producers

Producers are capable of producing multiple units of a single type of output good. At initialization, each producer is assigned a production capacity which specifies the maximum number of units each producer is able to produce of its output good.

In order to produce one unit of their output good, producers are required to acquire a number of units of each of their input goods equal to their input to output good ratio for that good. In order to produce two units of their output, producers require twice as many units of their inputs as for one good, and so on. Input to output good ratios are assigned to producers at initialization, and each producer may have different ratios for each of their input goods. Producers which do not require any inputs to produce their output good are known as no-input producers, and form the initial echelon of the supply chain. If a producer requires multiple types of input good, we refer to these goods as complementary. A producer cannot produce its output good unless it acquires all the necessary input goods.

Producers attach a reserve price R_p in producing their output good, which is a producer-specific constant. Reserve prices model the expense incurred by a producer in producing a single unit of their output, plus some small additional fixed profit margin. They therefore can be said to be equivalent to sale price of a single unit of the producer's output good. The total price charged by a producer is linear with the number of units of its output good that it produces: if a producer manufactures two units of its output good, it charges a price equal to $2R_p$, $3R_p$ if it produces three units, and so on.

B. Consumers

Consumers seek to acquire a number of units of their consumable good no greater than their desired consumable good quantity. In each network, each consumer is assigned a static consumption value V_c: this is the valuation the consumer holds for obtaining a single unit of its consumable good. Similar to reserve prices, the total value a consumer receives is linear with the number of goods a consumer obtains: if a consumer acquires two units of its consumable good it receives $2V_c$, if it acquires three units it receives $3V_c$ and so on.

C. Complementary Goods and Exposure

Some producers require multiple types of goods in order to produce their output good. This situation is referred to as an instance of input complementarity. Input complementarities, in the presence of resource contention, constrain the number of possible solutions to a supply chain network. Input complementarities also introduce the exposure problem in noncombinatorial approaches to SCF. The exposure problem occurs due to the fact that producers must acquire complementary goods individually. This leads to the risk of the producer acquiring some but not all of their required input goods. Combinatorial approaches avoid this problem by allowing

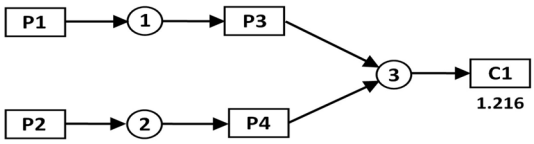

Fig. 2. Simple network from [1].

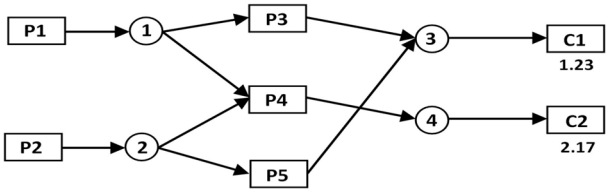

Fig. 3. Two-cons network from [1].

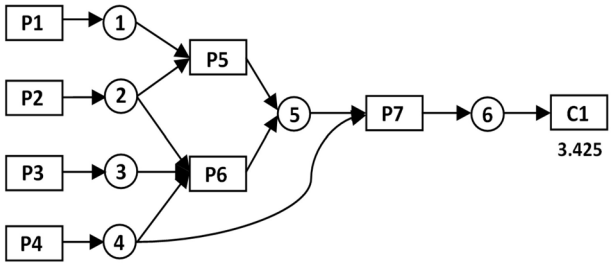

Fig. 4. Greedy-bad network from [1].

producers to bid on bundles of goods. Our LBP approach limits the exposure problem somewhat by encoding all of each producer's required inputs in each of their active states. The concept of states is explained in greater detail in Section IV-B.

D. Networks

For comparison and evaluation purposes we utilize a set of network structures used in [1], complemented by the huge network which we designed to show the performance of our mechanism on a very large network. In this section, we present these network structures and discuss their characteristics.

The Simple network, shown in Fig. 2, is a small three-tier network with two possible sources for the supply of C1's consumable good, good 3. Two-cons (Fig. 3), introduces the issue of complementary goods—P4 needs both goods 1 and 2 to produce its output. Because of this, only one of the consumers in this network can be satisfied at one time. The greedy-bad network (Fig. 4) introduces further complementarity issues. Producer P6 is a possible seller of one of Producer P7's input goods, good 5. However, in order to produce good 5, P6 requires good 4, which is also one of P7's inputs. As P7 is necessarily present in the single optimal solution to this network, it must buy good 5 from P5, even if the price is more expensive than when bought from P6. This network serves to show the weakness of greedy search-based techniques for SCF—although P7 may be able to acquire good 5 more cheaply from P6, in doing so it renders the rest of the supply chain infeasible.

Fig. 5 shows unbalanced, a larger network with several instances of complementarity. The many-cons network (Fig. 6) is a larger tree-structured network in which multiple consumers can be satisfied simultaneously. The Bigger network (Fig. 7) is large with many feasible solutions. Harder, shown in Fig. 8 can be seen as a much larger version of greedy-bad: despite the scale of this network, there exists only one possible solution due to the presence of a number of complementarities. Finally, the huge network (Fig. 9), models a very large-scale supply chain, with six tiers of production and three consumers.

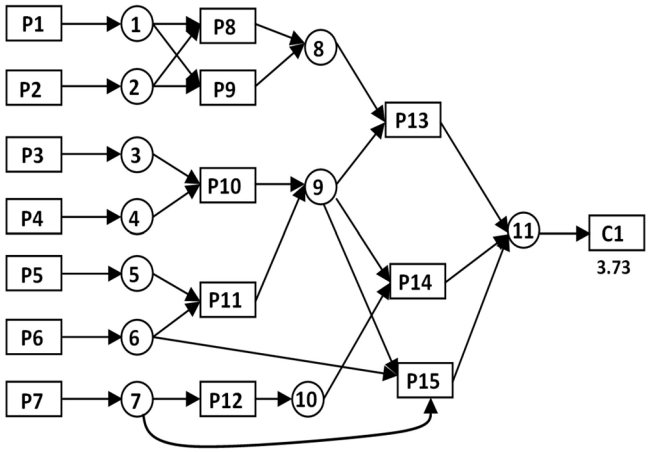

Fig. 5. Unbalanced network from [1].

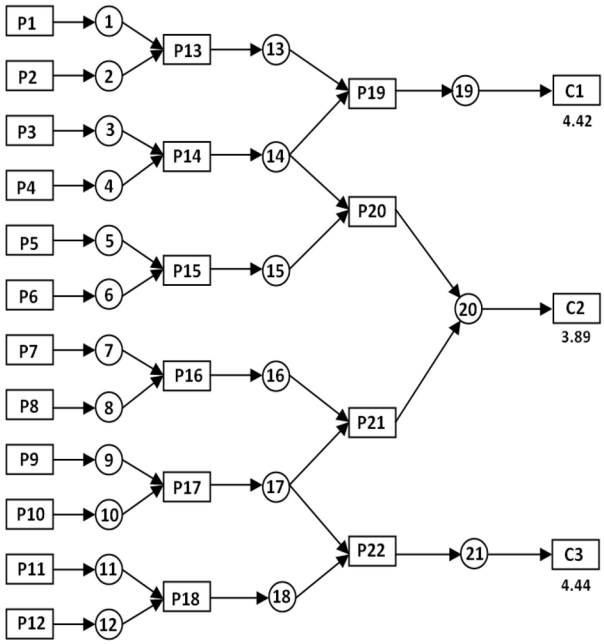

Fig. 6. Many cons network from [1].

E. Conversion to MRF Form

To convert task dependency networks into pairwise MRF form, two simple modifications must be made: first, the explicit representation of goods is removed from the network. Where edges previously linked an agent to a good or a good to an agent, edges now link agents directly, though they preserve the notion of an edge between agents meaning a potential route of exchange. Second, we remove direction from the edges from the graph. As an illustration, Fig. 10 depicts the MRF form of the supply chain network of Fig. 1. With the graph converted into pairwise MRF form, we define the states and costs required for the running of LBP.

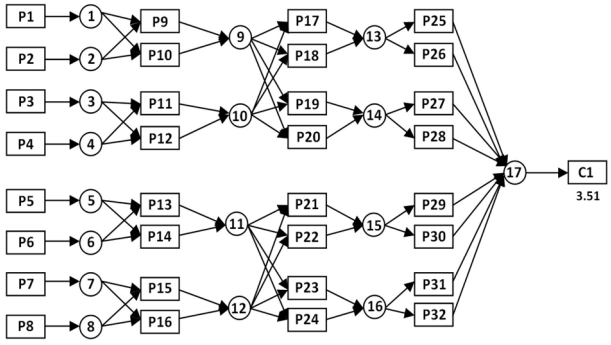

Fig. 7. Bigger network, from [1].

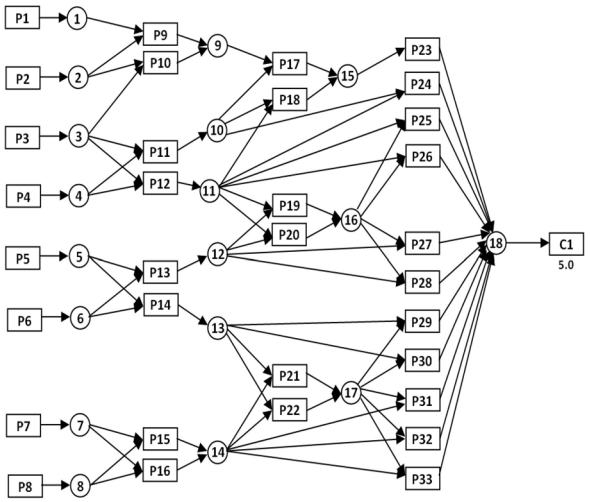

Fig. 8. Harder network, from [6].

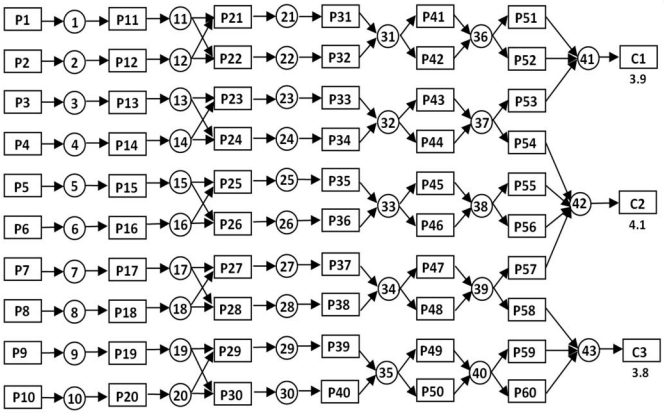

Fig. 9. Huge network.

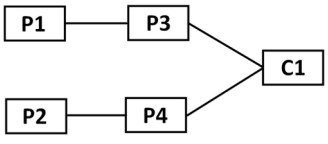

Fig. 10. Simple supply chain network converted into MRF form. Edges now link agents directly, and are undirected.

IV. Max-Sum Algorithm

The max-sum algorithm is a variant of LBP, a decentralized and distributed approximate inference scheme involving the application of Pearl's belief propagation algorithm [17] to graphical models containing cycles. It uses iterative stages of message passing as a means for estimating the MAP assignment. In our case, the MAP assignment is the network-wide state configuration that maximizes the value of (4). At each iteration of the max-sum algorithm, every node in the graph sends a message to each of its neighbors. Each message gives an estimation of the sender's beliefs about the potential cost to the total efficiency of the network by the states of the recipient. Nodes then update their beliefs about their own states based upon the content of the messages they received. The process of message passing and belief update continues until the beliefs of each node become stable. In Section IV-A, we discuss the properties of LBP while the rest of the section explains our use of the algorithm in this multiunit SCF scenario.

A. Properties of Max-Sum Algorithm

LBP is known to converge to exact results in a finite number of iterations on tree-structured graphs. While there is no such guarantee for more loopy graphs, and if convergence is reached, the solution will be an approximation, unless the graph contains only a single loop [19]. Reference [20] has established worst-case bounds on the quality of solutions produced by max-sum LBP, although these guarantees hold only when all unary and pairwise potentials are nonnegative, which is not the case in our model. Despite these limitations, LBP has seen great success in a number of areas, including turbo codes, low density parity check codes and stereo vision, as well as in communication in sensor networks [15], [24], and more recently, in supply chain emergence [2], [25]–[27].

B. States

In our implementation of the max-sum algorithm for supply chain emergence, for each agent there exists a finite number of purchases and sales (if the agent is a producer) in which the agent is viable, i.e., it acquires the necessary number of units of its input goods and sells the corresponding quantity of its output good. We encode each of these tuples of exchange relationships as states. For producers, each state defines a list of suppliers, a quantity bought from each supplier, a list of buyers and the quantity sold to each buyer. For consumers, a state lists a set of suppliers and the quantity bought from each supplier. For example, a possible state for producer P3 in Fig. 1 is "Buy 2 units of good 1 from P1 and sell to 1 unit of good 3 to C1." The number of states an agent possesses increases with the number of producers able to supply its input good(s), its production capacity or number of desired consumable good quantity, and the number of producers or consumers able to consume its output good. Because we model each type of good as an identical commodity, a producer or consumer can buy multiple units of the same good from different producers. For example, C1 derives equal value from acquiring 2 units of good 3 from Producer P2 and 1 unit of good 3 from Producer P3 as it does from acquiring 3 units of good 3 from either Producer P2 or P3. As well as a list of active states, we also allow for the inactive state, where the agent does not acquire or produce any goods.

C. Cost Function

We allow for two distinct types of cost, denoted as $f_v(x_v)$, the unary cost for agent v of being in state x_v, and $g_{uv}(x_u, x_v)$, the pairwise cost of connected agents u and v being in states x_u and x_v. LBP aims to find the allocation which minimizes these costs and thus provides the maximum surplus possible. Our cost function is given below

$$\epsilon(x_1, \ldots, x_N) = \sum_{v \in V} f_v(x_v) + \sum_{(u,v) \in E} g_{uv}(x_u, x_v). \quad (1)$$

$\epsilon(x_1, \ldots, x_N)$ is the set of agents, $f_v(x_v)$ is the unary cost of agent v being in state x_v, and $g_{uv}(x_u, x_v)$ is the pairwise cost of linked agents u and v, being labeled with states x_u and x_v. With all else equal, the lower the cost function result, the more efficient the allocation. We use the efficiency of the allocation as a measure of the quality of a solution.

D. Unary Cost

Each agent associates each of its states with a cost. For all agents, the cost of being in the inactive state is zero. For producers, all active states incur a positive cost, equal to the producer's reserve price R_p multiplied by the number of units the producer produces in that state. Consumers assign a negative cost to all states in which they acquire a good. If the consumer acquires a single unit of their consumable good in the state in question, the cost is equal to $0 - V_c$, where V_c represents the consumer's consumption value, the value the consumer assigns to the acquisition of its consumable good. If the consumer acquires two goods, the cost is equal to $0 - 2V_c$, with the cost decreasing linearly as the number of goods the consumer obtains increases.

E. Pairwise Cost

Pairwise costs encode the compatibility of two of the states of a pair of neighboring agents. Two states are compatible if agent i's state lists agent j as a buyer and the list of sellers in j's state includes i, and the number of units sold by i to j in i's state is equal to the number of units bought by j from i in j's state and vice versa. They are also compatible if agent i's state does not list agent j as a buyer and j's state does not list agent i as a seller and vice versa, or if both states are inactive states. If the states are compatible, the pairwise cost is equal to zero. If the two states do not meet either of these conditions, they are incompatible, and the pairwise cost of this combination of states is equal to positive infinity.

1) Belief Update: When updating their beliefs about their own states, agents add the unary cost of that state to the sum of the belief values contained within the messages passed from their neighbors in the previous step about that state. Thus, agent i's belief in its state x_u is calculated as

$$\text{bel}_i(x_u) = f_i(x_u) + \sum_{j \in N_u} m_{j \to i}(x_u) \quad (2)$$

where $f_i(x_u)$ is to the unary cost of i's state x_u, and $m_{j \to i}(x_u)$ are the messages received from i's set of neighbors $j \in N_u$ in the previous step.

2) Messages: At each iteration of LBP, each agent passes a message to each of its neighbors in the network containing a vector of belief values. These values correspond to the beliefs of the sender about the potential cost to the network of each of the recipient's states, and are calculated using

$$m_{i \to j}(x_v) = \min_{x_u} \Big(\text{bel}_i(x_u) - m_{j \to i}(x_u) + g_{ij}(x_u, x_v) \Big). \quad (3)$$

Each message contains a value for each of recipient j's states. To calculate the value to be sent for state x_v of j, for each of it's own states x_u, i subtracts the pairwise cost of x_u plus the belief value in message passed in the previous step from j about x_u from i's current belief in state x_u. This process continues until all of i's states have been compared with j's state x_v, at which point the minimum of these values is added to the message, and the process is repeated for the next of j's states. Once values have been calculated for all of j's states, the message is passed from i to j.

F. Allocation

Before allocation is performed, each agent determines their final state—the state, at convergence, in which the agent believes holds the lowest cost. Once the final states of each of the agents have been determined, agents share their final states with their neighbors and the process of allocation begins. Each agent removes edges to other agents not listed in their final state, and agents with an inactive final state remove all of their edges. They remove edges leading to neighbors who do not mutually list them in their final state, or with whom there is a mutual listing but a misalignment in terms of quantities. If, once edges are removed, a producer is left with edges to suppliers enabling it to acquire each its input goods in the desired quantities as well as one or more edges to buyers then it is regarded as active in the allocation. Similarly, consumers left with edges leading to suppliers of their consumable good are designated as active in the allocation.

G. Allocation Value

We calculate allocation values using (4). We sum the products of the consumption value V_c of each active consumer c in the set of active consumers C and the number of goods obtained by c, A_c. From this, we subtract the sum of the products of the reserve price R_p of each active producer p in the set of active producers P and the number of goods manufactured by p, M_p to produce a final allocation value

$$\text{Val} = \sum_{c \in C} V_c A_c - \sum_{p \in P} C_p M_p. \quad (4)$$

H. Payments

An active producer p is paid a fee by each of the buyers of its output good, at a price equal to the value of

$$F_{bp} = R_p M_{pb} + \frac{M_{pb}}{M_p} \sum_{i \in SP} F_{pi}. \quad (5)$$

F_{bp} is the fee paid from buyer b to producer p, R_p is p's reserve price, M_{pb} is the number of goods manufactured by p for buyer b, M_p is the total number of goods produced by

producer p, F_{pi} is the fee paid by p to each supplier i from p's set of suppliers SP. Buyers pay p's marginal cost of producing each good they purchase plus a proportion of p's input good costs commensurate with the proportion of p's total output they purchase. No payments are made to or by the mechanism.

I. Convergence

We make use of a convergence detector agent, as recommended in [1] for scenarios with multiple agents in initially nonquiescent states, which controls termination but is otherwise uninvolved in the workings of the algorithm. The use of such an agent is not unrealistic for SCF settings—in the real world, it is highly likely that SCF facilitation would be provided by a coordinating third party, who would be represented by such an agent, rather than being instituted directly by the businesses themselves. This means that limited communication between participating agents and the third party is a reasonable assumption. Indeed, this assumption of communication with third parties is made by all market-based approaches, whether centralized or decentralized, in the form of bids placed in auctions or negotiation via dedicated mediator agents.

Even in decentralized approaches such as [1], the market for a single good can ultimately be viewed as centralized, with a single auction responsible for aggregating the bids of multiple participants and deciding the winning bidders. The difference between this approach and a centralized approach is that the solution produced relies upon the result of multiple local auctions, rather than one single global auction. Thus, the goal in decentralized SCF is not to produce a technique for the niche situation which requires all aspects of the process to be completely decentralized, but to decentralize the most important aspect of the process, which is to compute global solutions in a local manner on the basis of limited information. Our convergence detector agent detects convergence more quickly and with less overhead than a distributed algorithm for acyclic directed graphs (e.g., the Dijkstra–Scholten [28], which involves a stage of spanning tree construction). It does not, however, affect the computation of solutions and thus does not centralize our approach.

It is also important to note that while the use of a convergence detector agent serves to shorten the running time of the algorithm, the fact that it is not required for the algorithm to produce solutions means that it does not represent a single point of failure. This in contrast with the auctions or mediators present in market based approaches, the failure of which would in all centralized approaches and the majority of decentralized approaches—if the failed auction is for a critical good—prevent the technique from producing solutions.

Once the LBP algorithm has begun, each agent reports to the convergence detector agent at each iteration specifying whether the state in which they believe holds the lowest cost has changed since the previous iteration. If the current number of iterations is greater than the size of the spanning tree—as is explained in the following paragraphs—and all agents reported that their lowest-cost state has remained the same as the previous iteration, then the convergence detector agent halts the algorithm, and allocation is performed. The process of allocation is outlined in Section IV-F.

As mentioned in Section IV-A, LBP is known to converge on tree-structured graphs in a number of iterations equal to the graph's diameter. Although not all of our networks are trees, we take this value as the earliest number of iterations at which it can be said that LBP has converged. In the absence of an efficient, distributed and general technique for finding an exact value for the graph diameter, we use distributed depth-first search, according the method proposed in [29], to find a spanning tree of the graph. The diameter of the spanning tree is then determined using the distributed Bellman–Ford algorithm in order to provide an upper bound for the value of the actual diameter of the graph.

It is important to note that while the use of a convergence detector agent serves to shorten the running time of the algorithm, a random agent present in the original network can instead perform the function of the convergence detector agent if a fully decentralized approach is required. We refer to such an agent as the "coordinator agent." Once LBP has reached a number of iterations equal to the diameter of the spanning tree, the coordinator agent initiates a distributed breadth-first search similar to that used to find the diameter of the spanning tree. This time, agents send messages to their neighbors indicating whether they have reached convergence or not. These messages are propagated back to the coordinator agent. Once the coordinator agent has received a message indicating the convergence status of each node in the network—it is aware of the identities of each agent, though not their costs or capabilities, through the construction of the spanning tree—then it either terminates the algorithm if all agents have converged, or allows it to continue for a number of steps equal to the diameter of the spanning tree.

V. Experiments

We perform two sets of experiments, examining the efficiency of the allocations produced by LBP in both static and dynamic environments. We compare results in the static case to multiunit reimplementations of the SAMP-SB and SAMP-SB-D auction protocols presented in [1]. Due to the absence of a comparable technique for our dynamic environment experiments, we compare these results with those produced by LBP in the static case.

A. Static Environment Experiments

In the static environment, we compare LBP with multiunit implementations of two decentralized auction protocols from [1]. We test each technique over 100 runs on each of the supply chain networks given in [1]. We extended SAMP-SB and SAMP-SB-D according to the suggestion proposed in [1] by using multiple copies of each agent to represent each unit of a given producer's capacity or consumer's number of desired goods. Because this representation does not allow for the use of input to output good ratios, for fair comparison we also test LBP with all ratios set to 1, which we refer to as ratioless LBP. Ratios serve to constrain the number of possible solutions in the network. We perform 100 runs of ratioless LBP, standard multiunit LBP, SAMP-SB and SAMP-SB-D for each network, varying input ratios and consumer desired goods

TABLE I
FULL LIST OF THE ALTERATIONS WE ALLOW FOR

Category	Alteration Type
New Entrant	Producer Added
New Entrant	Consumer Added
Departure	Producer Removed
Departure	Consumer Removed
Property Change	Reserve Price
Property Change	Input:Output Goods Ratio
Property Change	Production Capacity
Property Change	Consumption Value
Property Change	Consumer Desired Goods

(for LBP) as well as reserve prices and production capacities (for all methods) between each run. We discard runs in which the optimal allocation value, determined using mixed integer programming, is nonpositive. Reserve prices are drawn from the distribution $U(0, 1)$, production capacities from the interval $(4, 5)$, consumer desired goods from $(2, 3)$, and input to output good ratios from $(1, 2)$. We use these fairly large values for production capacities and consumer desired goods and fairly low values for input to output good ratios to allow for feasibility in larger networks; however, as long as a positive-valued solution exists, the performance of LBP is largely unaffected by the values used. Consumption values are fixed at the per-network values given in [1] over every run.

B. Dynamic Environment Experiments

We perform three sets of experiments for each network in the dynamic environment, testing LBPs ability to deal with alterations to the structure of each network and to the properties the participants within to various extremes. Assessing LBPs performance in these scenarios is important because the changes to beliefs brought about as a result of these alterations have the potential to disrupt the proper functioning of the algorithm, while the performance of ascending auction-based approaches such as SAMP-SB are inherently unaffected by these changes. We describe these settings in Section V-F. For all experiments, initial producer reserve prices are drawn from the distribution $U(0, 1)$, input good ratios from the interval $(1, 2)$, capacities from $(4, 5)$ and consumer desired goods from the interval $(2, 3)$. A full list of the alterations we allow for is shown in Table I. Parameters specific to changes involving new entrants are explained in the following subsections.

C. New Entrants

In the interest of offering a mechanism capable of rapid response to new entrants into the market, we allow for the possibility of new producers and consumers entering the process at any point before the final allocation is determined. For the purposes of our experiments, we assign a tier value to each good in the original, unaltered network. Goods produced by producers with no inputs are tier 1 goods, goods produced by producers which consume tier 1 goods are tier 2 goods, and so on. For the purposes of these experiments, when a new producer enters the market, it is assigned a randomly chosen output good and an appropriate set of input goods from the tier below its output good. The number of inputs each producer is assigned is drawn randomly from $(1, G_t)$ where G_t is the total number of goods in the appropriate tier. If the producer is assigned a tier 1 output good, it is given no inputs. We assign input goods in this way in order to avoid scenarios where producers are able to bypass multiple echelons of the original supply chain by processing a low-tier good into an output from a much higher tier, which simplifies the problem of determining the optimal allocation, and also to prevent producers from producing outputs of a lower tier than their input goods, which is unrealistic. Producers are also assigned a random reserve price, input to output good ratios and a capacity, at the values specified in Section V-A.

Consumers are randomly assigned a consumable good from a uniform distribution of the goods in the final tier of the original network. For the purposes of our experiments, consumption values for new consumers are set at a value randomly drawn from a uniform distribution of the values plus or minus 10% of the consumption value of consumer C1 in the original network. We set consumption values for new consumers within these bounds so that the problem is not oversimplified by the entrance of consumers with very high consumption values. Conversely, consumers with very low consumption values would be unlikely to acquire their consumable goods. The number of goods new consumers desire is set in the same way as for consumers present at initialization. The value used for this property can be found in Section V-A.

D. Departures

If a producer or consumer wishes to leave the market, it notifies each of its neighbors that it intends to leave, and then removes all of its edges and states. Neighboring agents then remove any states which list the departee as a buyer or seller. In order to prevent manipulation of the mechanism, once a participant has left the process it is not allowed to rejoin.

E. Property Changes

As shown in Table I, we allow for the alteration of each of the following properties: producers may change their reserve price, their input to output good ratios and production capacities, while consumers may change their consumption values and their number of desired goods.

F. Dynamic Environment: Experimental Settings

In the following subsections, we explain the details of each of our three experimental settings.

G. Dynamic Environment Setting 1: Single Change, Single Type

In the first set of dynamic experiments we run, we test LBPs ability to cope with a single incidence of each of the nine possible changes listed in Table I. This scenario tests LBPs ability to cope with minor alterations to the original network. We perform 100 runs of LBP on each network for each change. In each run, a random step is selected between step 1 and AD,

the approximate graph diameter, with equal likelihood for each step, as determined by the convergence detector agent. LBP continues to run until convergence is detected, as is specified in Section IV-I, or, if convergence is not reached, until the algorithm has run for 250 steps.

H. Dynamic Environment Setting 2: Multiple Changes, Multiple Types

In the second set of dynamic experiments, we test LBPs performance under a series of random changes. This allows us to assess LBPs performance under more challenging conditions than in Setting 1. The number of changes for each run is drawn from the interval $(2, 5)$ and is varied between runs. The step at which the first change occurs follows the same method as in Setting 1. The steps at which each of the subsequent changes occur are drawn from the interval (C_{prev}, AD), where C_{prev} is the step at which the previous change occurred and AD is the approximate graph diameter. These values are recomputed for every run. For this and each subsequent set of experiments, once the final change occurs, LBP continues to runs until convergence or until 250 steps have been completed.

I. Dynamic Environment Setting 3: More Changes, Multiple Types

Finally, the third set of dynamic experiments model a scenario similar to experimental setting 2, but this time allow for the possibility of a larger number of random alterations. This tests LBPs ability to cope under very challenging conditions, with numerous changes being made during the running of the algorithm. In this set of experiments, the number of changes per run is drawn from the interval $[6 \ldots 10]$, and is varied between runs.

We model a dynamic environment by removing and adding participants, and changing the properties of existing participants. The potential alterations we allow for can be grouped into three categories—new entrants, departures and property changes. A full list of possible alterations, sorted by category is shown in Table I. We explain the details of each of these three categories in the following subsections.

J. Performance Evaluation

In both the static and dynamic environments, we run LBP on each network until a convergent state is reached, using the value of the allocations produced as a measure of the quality of our solutions. If LBP does not converge before 50 iterations, we record the result as a zero-valued allocation, which indicates that no solution was found. We use lp_solve, a free mixed integer programming solver which computes the optimal allocation in a centralized manner, to provide a benchmark optimal allocation value to compare LBP against. We present our results in the form of average efficiency, which is calculated by dividing the total allocation values produced by each method over 100 runs on each network by the maximum available value, as determined by lp_solve, over the same 100 runs. An average efficiency of 1.000 indicates that 100% of the available value was captured on each of the hundred runs for that particular network instance, and represents the best possible result. Allocation values are calculated as per Section IV-G.

TABLE II
AVERAGE EFFICIENCY IN EACH NETWORK PRODUCED BY THE PROPOSED LBP-BASED TECHNIQUE, AND THE SAMP-SB AND SAMP-SB-D PROTOCOLS FROM [1]. A RESULT OF 1.000 IS EQUAL TO THE CAPTURE OF AN AVERAGE OF 100% OF AVAILABLE EFFICIENCY, WHILE A RESULT OF −1.000 IS EQUAL TO AN AVERAGE CAPTURE OF −100% OF AVAILABLE EFFICIENCY. NOTE THAT WHILE 1.000 IS THE MAXIMUM ACHIEVABLE POSITIVE VALUE, IT IS POSSIBLE TO PRODUCE NEGATIVE OVERALL EFFICIENCIES BELOW −1.000. THE BIGGER NETWORK WITH RATIOS PRODUCED THE LARGEST STANDARD DEVIATION WITH A VALUE OF 0.12, INDICATING VERY LITTLE VARIABILITY WITHIN OUR RESULTS

Network	Average Efficiency			
	LBP no ratios	LBP with ratios	SAMP-SB	SAMP-SB-D
Simple	1.000	1.000	0.999	0.999
Unbalanced	0.962	0.872	0.964	0.998
Two-Cons	0.986	0.983	0.963	0.998
Bigger	0.969	0.813	0.995	1.000
Many-Cons	1.000	1.000	0.425	0.997
Greedy-Bad	0.91	0.839	0.666	0.923

TABLE III
PER-NETWORK COMPARISON OF THE AVERAGE NUMBER OF MESSAGES PASSED BEFORE CONVERGENCE IN MULTIUNIT LBP WITH AND WITHOUT RATIOS COMPARED TO THE AVERAGE NUMBER OF BIDS PLACED BEFORE QUIESCENCE IN THE SAMP-SB PROTOCOL FROM [1]

Network	LBP no ratios avg number of messages passed	LBP with ratios avg number of messages passed	SAMP-SB avg number of bids placed	SAMP-SB avg number of bids & price quotes sent
Simple	51.76	52.96	210.64	2419.49
Unbalanced	494.96	518.42	1201.21	21098.79
Two-Cons	106.54	108.64	905.38	12009.91
Bigger	2859.84	2964.96	2239.43	71276.2
Many-Cons	498.72	488.64	5797.36	68017.69
Greedy-Bad	118.62	119.7	1406.37	18379.73

VI. RESULTS

A. Static Environment: Average Efficiency

Table II shows the average efficiency produced by each method—multiunit LBP without ratios, multiunit LBP with ratios, and our multiunit implementations of the SAMP-SB and SAMP-SB-D auction protocols. Input to output good ratios are not present in SAMP-SB and SAMP-SB-D. We see that both LBP-based methods are able to match or outperform SAMP-SB on the majority of networks, whilst also matching the results produced by SAMP-SB-D on many network instances. As expected, LBP finds the optimal allocation 100% of the time on acyclic networks, while still being able to produce highly efficient allocations on more loopy networks. We also see that LBP tended to perform better when input to output good ratios are not present; this is to be expected since the presence of ratios serves to constrain the number of solutions available.

B. Static Environment: Messages/Bids Before Convergence

From Table III, we see that the LBP methods tend to require far fewer exchanges of information to converge to a

TABLE IV
DISTRIBUTION OF EFFICIENCY CLASSES PRODUCED BY LBP UNDER EXPERIMENTAL SETTING 1, WHERE A CHANGE OF A SINGLE TYPE OCCURS ONCE PER RUN

	Simple network % of instances				Unbalanced network % of instances				Two-Cons network % of instances				Bigger network % of instances			
Change	Neg	Zero	Sub	Opt	Neg	Zero	Sub	Opt	Neg	Zero	Sub	Opt	Neg	Zero	Sub	Opt
Production Cost	0.0	1.0	0.0	99.0	1.0	12.0	22.0	65.0	3.0	0.0	2.0	95.0	4.0	2.0	0.0	94.0
Ratio	1.0	3.0	0.0	96.0	1.0	6.0	14.0	79.0	3.0	0.0	1.0	96.0	4.0	0.0	1.0	95.0
Consumption Value	0.0	0.0	0.0	100.0	0.0	7.0	23.0	70.0	0.0	0.0	4.0	96.0	2.0	0.0	0.0	98.0
Producer Added	0.0	4.0	0.0	96.0	4.0	17.0	34.0	45.0	3.0	0.0	18.0	79.0	6.0	15.0	4.0	75.0
Consumer Added	10.0	1.0	8.0	81.0	4.0	0.0	52.0	44.0	11.0	0.0	9.0	80.0	16.0	0.0	2.0	82.0
Producer Removed	0.0	0.0	0.0	100.0	0.0	10.0	8.0	82.0	0.0	0.0	0.0	100.0	3.0	0.0	0.0	97.0
Consumer Removed	0.0	0.0	0.0	100.0	0.0	0.0	0.0	100.0	0.0	0.0	0.0	100.0	0.0	0.0	0.0	100.0
Production Capacity	0.0	1.0	0.0	99.0	1.0	9.0	18.0	72.0	2.0	1.0	0.0	97.0	1.0	2.0	0.0	97.0
Desired Goods	0.0	0.0	0.0	100.0	1.0	5.0	15.0	79.0	6.0	0.0	0.0	94.0	9.0	0.0	0.0	91.0

TABLE V
DISTRIBUTION OF EFFICIENCY CLASSES PRODUCED BY LBP UNDER EXPERIMENTAL SETTING 1, WHERE A CHANGE OF A SINGLE TYPE OCCURS ONCE PER RUN

	Many-Cons network % of instances				Greedy-Bad network % of instances				Harder network % of instances				Huge network % of instances			
Change	Neg	Zero	Sub	Opt	Neg	Zero	Sub	Opt	Neg	Zero	Sub	Opt	Neg	Zero	Sub	Opt
Production Cost	0.0	0.0	0.0	100.0	0.0	14.0	15.0	71.0	44.0	26.0	2.0	28.0	4.0	21.0	12.0	63.0
Ratio	0.0	0.0	0.0	100.0	0.0	4.0	18.0	78.0	46.0	24.0	3.0	27.0	7.0	27.0	7.0	59.0
Consumption Value	0.0	0.0	0.0	100.0	0.0	13.0	10.0	77.0	57.0	12.0	1.0	30.0	8.0	21.0	11.0	60.0
Producer Added	1.0	4.0	9.0	86.0	0.0	18.0	28.0	54.0	52.0	25.0	4.0	19.0	3.0	29.0	9.0	59.0
Consumer Added	0.0	0.0	0.0	100.0	0.0	1.0	28.0	71.0	51.0	17.0	16.0	16.0	3.0	29.0	9.0	59.0
Producer Removed	0.0	0.0	0.0	100.0	0.0	14.0	0.0	86.0	38.0	24.0	5.0	33.0	10.0	17.0	9.0	64.0
Consumer Removed	0.0	0.0	0.0	100.0	0.0	0.0	0.0	100.0	0.0	0.0	0.0	100.0	1.0	25.0	1.0	73.0
Production Capacity	0.0	0.0	0.0	100.0	0.0	12.0	20.0	68.0	50.0	22.0	3.0	25.0	8.0	23.0	10.0	59.0
Desired Goods	0.0	0.0	0.0	100.0	0.0	9.0	14.0	77.0	51.0	18.0	3.0	28.0	1.0	27.0	16.0	56.0

solution than is required in the auction-based methods. The number of bids and price quotes exchanged in SAMP-SB and SAMP-SB-D are identical—the only difference between the protocols is a post-allocation decommitment stage—so we do not include values for SAMP-SB-D in this table. It is also clear that despite the differences in average efficiency between LBP with ratios and LBP without ratios, both methods converge at roughly the same point in all networks. When price quotes are taken into account, the frequency of information exchange in SAMP-SB tends to be orders of magnitude greater than in LBP, offsetting the fact LBP messages tend to encode more information and thus are of a larger size than bids in SAMP-SB.

C. Dynamic Environment Setting 1: Single Change, Single Type

Tables IV and V show the efficiency classes produced by LBP over 100 runs of each alteration type over each network. Given the absence of a comparable decentralized supply chain reconfiguration technique in the literature, the main basis of comparison with these results is with those of multiunit LBP with ratios.

In the Simple network, for which multiunit LBP with ratios achieved 100% optimality, the results of Table IV suggest that the effect of most alterations is reasonably minor, with at most 4% of these optimal results being converted into zero or negative-valued allocations for eight of the nine alterations we tested. The sole exception to this is when a new consumer is added to the network: in this case, 20% of the original optimal results are lost to zero or negative-valued allocations. Efficiency loss when a new consumer added is a common occurrence on many of our networks; this is because the addition of new vertices and edges to the original graph often also leads to the creation of one or more cycles. As we explain in Section II-B3, there are no performance guarantees for LBP on graphs with multiple cycles.

For the unbalanced network, for which multiunit LBP with ratios produced optimal results 67% of the time in a non-reconfiguration setting with a relatively large proportion of suboptimal results, we see that reconfiguration actually leads to a slight improvement in the proportion of optimal results in several instances. We speculate that the reason for this might be that certain changes, such as a producer being removed, may lead to the removal of cycles or a widening of the difference in efficiency between optimal and suboptimal solutions, making it easier for LBP to find optimal solutions. We also notice that LBP is able to deal with the removal of the sole consumer in the network and consistently produces optimal

TABLE VI
AVERAGE EFFICIENCY FOR EACH TYPE OF CHANGE PRODUCED BY LBP IN EXPERIMENTAL SETTING 1. A RESULT OF 1.000 IS EQUAL TO THE CAPTURE OF AN AVERAGE OF 100% OF AVAILABLE EFFICIENCY

	Network							
Change	Simple	Unbalanced	Two-Cons	Bigger	Many-Cons	Greedy-Bad	Harder	Huge
Production Cost	0.996	0.868	0.926	0.767	1.000	0.863	-0.022	0.576
Ratio	0.976	0.860	0.93	0.808	1.000	0.930	-0.310	0.382
Consumption Value	1.000	0.859	0.973	0.904	1.000	0.882	-0.496	0.413
Producer Added	0.954	0.709	0.812	0.752	0.886	0.824	-0.556	0.576
Consumer Added	0.747	0.720	0.724	0.670	1.000	0.917	0.19	0.581
Producer Removed	1.000	0.855	1.000	0.857	1.000	0.824	-0.227	0.308
Consumer Removed	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.621
Production Capacity	0.985	0.838	0.966	0.932	1.000	0.845	-0.715	0.425
Desired Goods	1.000	0.909	0.871	0.691	1.000	0.908	-0.466	0.592

allocations (of value zero) for this alteration on this and all other networks with a single consumer.

Results for all other networks—Two-Cons (98% optimal results in the nonreconfiguration multiunit case), bigger (94%), many-cons (100%), greedy-bad (69%), harder (27%), and huge (63%)—follow in a similar vein, with most alterations tending not to significantly change the proportion of optimal results produced. Certain alterations, in particular the removal of producers and consumers, appear to consistently improve the results LBP is able to produce for most networks. Others, most notably the addition of producers, tend to mean LBP is able to produce slightly fewer optimal allocations. Again, we attribute improvements in performance brought about by departures of producers to the removal of cycles leading to less frequent double-counting of messages and more accurate agent beliefs. Conversely, new producers and consumers joining the network tend to reduce performance slightly by introducing additional cycles to the networks.

Average efficiency results for experimental setting 1 are presented in Table VI, and follow a similar pattern to the efficiency classes results. For all but the Harder network, average efficiencies tend to be roughly similar to the values for multiunit LBP with ratios in a nonreconfiguration setting, as is shown in Table II. Again, adding a producer seems to have the most pronounced adverse effect, with efficiency loss of around 25% for most networks when this alteration is made.

D. Dynamic Environment Setting 2: Multiple Changes, Multiple Types

Table VII shows the efficiency classes produced in experimental setting 2, where randomly chosen alterations occur between 2 and 5 times per run.

Because removing a consumer leads to 100% optimality in most networks, and negates the effect of further shocks, we do not allow this alteration to be chosen as a random shock in this setting or in experimental setting 3. A set of 100 runs were performed for each network. Table VIII shows the average efficiency produced by LBP in this setting. It is clear from these results that multiple alterations have little effect on LBPs performance, with only very slight degradations in total optimality for most networks.

TABLE VII
DISTRIBUTION OF EFFICIENCY CLASSES PRODUCED BY LBP UNDER EXPERIMENTAL SETTING 2, WHERE RANDOMLY CHOSEN CHANGES OCCUR BETWEEN 2 AND 5 TIMES PER RUN

	LBP % of instances			
Network	Neg	Zero	Sub	Opt
Simple	1.0	1.0	1.0	97.0
Unbalanced	1.0	10.0	28.0	61.0
Two Cons	5.0	0.0	5.0	90.0
Bigger	13.0	4.0	2.0	81.0
Many-Cons	2.0	2.0	2.0	94.0
Greedy-Bad	0.0	11.0	21.0	68.0
Harder	53.0	18.0	6.0	23.0
Huge	7.0	16.0	16.0	60.0

TABLE VIII
AVERAGE EFFICIENCY FOR EACH TYPE OF CHANGE PRODUCED BY LBP IN EXPERIMENTAL SETTING 2. A RESULT OF 1.000 IS EQUAL TO THE CAPTURE OF AN AVERAGE OF 100% OF AVAILABLE EFFICIENCY

Network	LBP average efficiency
Simple	0.954
Unbalanced	0.806
Two Cons	0.822
Bigger	0.528
Many-Cons	0.940
Greedy-Bad	0.818
Harder	-0.118
Huge	0.511

E. Dynamic Environment Setting 3: Many Changes, Multiple Types

Table IX shows the efficiency classes produced in experimental setting 3, while Table X shows the average efficiency for the same experimental setting. In this setting, randomly chosen alterations occur between 6 and 10 times per run, simulating a SCF scenario requiring a great deal of adaptation by the mechanism. We allow for all of the alterations listed in

TABLE IX
DISTRIBUTION OF EFFICIENCY CLASSES PRODUCED BY LBP UNDER EXPERIMENTAL SETTING 3, WHERE RANDOMLY CHOSEN CHANGES OCCUR BETWEEN 6 AND 10 TIMES PER RUN

	LBP % of instances			
Network	Neg	Zero	Sub	Opt
Simple	2.0	0.0	3.0	95.0
Unbalanced	3.0	11.0	29.0	57.0
Two Cons	4.0	1.0	8.0	87.0
Bigger	14.0	5.0	7.0	74.0
Many-Cons	0.0	2.0	2.0	96.0
Greedy-Bad	1.0	17.0	13.0	69.0
Harder	55.0	21.0	5.0	19.0
Huge	20.0	14.0	20.0	46.0

TABLE X
AVERAGE EFFICIENCY FOR EACH TYPE OF CHANGE PRODUCED BY LBP IN EXPERIMENTAL SETTING 3. A RESULT OF 1.000 IS EQUAL TO THE CAPTURE OF AN AVERAGE OF 100% OF AVAILABLE EFFICIENCY

Network	LBP average efficiency
Simple	0.911
Unbalanced	0.713
Two Cons	0.801
Bigger	0.520
Many-Cons	0.989
Greedy-Bad	0.793
Harder	-0.074
Huge	0.379

Table I with the exception of the departure of consumers. As previously stated, this is because the departure of a consumer reliably leads to 100% optimality in most networks. When compared to the results of experimental setting 2, we see that LBPs performance degrades gracefully with a larger number of changes per run. Total optimality and average efficiency are very slightly worse that experimental setting 2 and, for most networks, are roughly comparable to the results produced when no reconfiguration is performed at all.

F. Game-Theoretic Properties

In this section, we analyze the game-theoretic properties of our LBP-based approach, and compare them to those of SAMP-SB and SAMP-SB-D.

1) Individual Rationality: A mechanism is classified as individually rational if a participant cannot receive negative utility by participating. As with SAMP-SB, we cannot guarantee the individual rationality of our approach given that there exists the possibility of dead ends being present in our allocations. Individual rationality could be guaranteed in our approach using a process of post-allocation decommitment similar to that used by SAMP-SB-D.

2) Incentive Compatibility: A mechanism is incentive compatible if the dominant strategy for participants is to truthfully reveal their private valuations. At present, our mechanism is not incentive compatible because participants may potentially increase their utility by inflating their reserve prices. However, there is an uncertain upper limit to this potential increase in utility; if a producer reports a reserve price which is too high, there may be an alternative, cheaper allocation in which the misreporting agent does not participate. This limit is uncertain as our agents operate under limited local information. This is an issue for sellers in any real-life market-based scenario.

3) Budget Balance: Our approach involves no payments either to or from the mechanism, and is therefore strongly budget balanced. This property is also present in SAMP-SB/D, as no payments are made to or by the auctions.

4) Allocative Efficiency: In static scenarios, LBP guarantees perfect allocative efficiency on acyclic networks due to its ability to reliably converge to the optimal MAP assignment. If LBP converges on a network with a single loop, the resulting allocation is also guaranteed to have perfect efficiency. Because there is no guarantee of the quality of solutions produced by LBP on networks with more than a single loop, allocative efficiency for these networks is also impossible to guarantee. Encouragingly, LBP showed strong allocative efficiency on all of the loopy networks we tested. It is also important to note that other techniques for SCF tend to have difficulties under certain conditions: SAMP-SB tends to perform poorly under certain sets of producer reserve prices (specifically, prices which do not permit competitive equilibrium—see [1] for further details) and combinatorial approaches may face problems with scalability.

VII. CONCLUSION

Decentralized approaches for SCF provide advantages over centralized techniques in terms of scalability and an absence of a single point of failure. However, existing decentralized techniques have tended to be applied to abstract scenarios where goods are exchanged one unit at a time. In this paper, we presented a technique for decentralized multiunit SCF using max-sum LBP. We tested the performance of LBP in both static and dynamic environments. In a static environment, our results indicate that LBP outperforms SAMP-SB; it also performs comparably to SAMP-SB-D without performing any post-allocation decommitment. LBP also requires much less information to be exchanged than either of the auction protocols. Our dynamic environment experiments allowed us to evaluate LBPs performance in a setting where the algorithm is forced to adapt to changes in the properties or composition of participating agents. We tested the effect on efficiency produced by each individual change, as well as the effect produced by multiple successive randomly-chosen changes. We found that LBPs performance remains solid when faced with the majority of the changes we applied, and that performance degrades gracefully in the presence of large numbers of alterations.

An important characteristic of the LBP mechanism for decentralized supply chain emergence is that, unlike the auction-based techniques, it is not a market-based approach. While messages in LBP resemble bids in an auction, the information they carry is significantly richer than just price. This

allows the emergence of chains that not only are efficient in terms of cost, but also take into consideration production capacities (and can be enriched to deal with further constraints such as quality, deadlines, reputation) in a consistent and efficient manner.

In future work, we intend to focus on algorithms to unfold loops, potentially improving performance on loopy networks. The properties of goods could be expanded to model factors such as quality and delivery dates. Our work on dynamic response to changes in the structure of the supply chains and the properties of the participants could be combined with recent work on reliability of networks [30] to provide a measure on how resilient a particular formation is. Additionally, because we at present assume all agents are truth-telling, allowing for strategic behavior in the form of misrepresentation of beliefs in outgoing messages presents an interesting avenue to pursue.

REFERENCES

[1] W. Walsh and M. Wellman, "Decentralized supply chain formation: A market protocol and competitive equilibrium analysis," *J. Artif. Intell. Res.*, vol. 19, pp. 513–567, Nov. 2003.

[2] M. Winsper and M. Chli, "Decentralized supply chain formation using max-sum loopy belief propagation," *Comput. Intell.*, vol. 29, pp. 280–309, May 2012.

[3] D. Pardoe and P. Stone, "TacTex-05: A champion supply chain management agent," in *Proc. 21st Nat. Conf. Artif. Intell. (AAAI)*, Boston, MA, USA, 2006, pp. 1489–1494.

[4] M. Wellman *et al.*, "Strategic interactions in a supply chain game," *Comput. Intell.*, vol. 21, no. 1, pp. 1–26, 2005.

[5] J. Collins *et al.*, "The supply chain management game for the 2007 trading agent competition," School Comput. Sci., Carnegie Mellon Univ. Pittsburgh, PA, USA, Tech. Rep. CMU-ISRI-07-100, 2006.

[6] W. Walsh and M. Wellman, "Modeling supply chain formation in multiagent systems," in *Agent Mediated Electronic Commerce II*. Berlin, Germany: Springer, 1999.

[7] R. B. Myerson, "Efficient mechanisms for bilateral trading," *J. Econ. Theory*, vol. 29, no. 2, pp. 265–281, Apr. 1983.

[8] M. Babaioff and W. Walsh, "Incentive-compatible, budget-balanced, yet highly efficient auctions for supply chain formation," *Decis. Support Syst.*, vol. 39, no. 1, pp. 123–149, 2003.

[9] W. Walsh, M. Wellman, and F. Ygge, "Combinatorial auctions for supply chain formation," in *Proc. 2nd ACM Conf. Electron. Commerce (EC)*, Minneapolis, MN, USA, 2000, pp. 260–269.

[10] M. Babaioff and N. Nisan, "Concurrent auctions across the supply chain," in *Proc. 3rd ACM Conf. Electron. Commerce (EC)*, New York, NY, USA, pp. 1–10, 2001.

[11] J. Cerquides, U. Endriss, A. Giovannucci, and J. A. Rodríguez-Aguilar, "Bidding languages and winner determination for mixed multi-unit combinatorial auctions," in *Proc. Int. Joint Conf. Artif. Intell.*, 2007, pp. 1221–1226.

[12] B. Ottens and U. Endriss, "Comparing winner determination algorithms for mixed multi-unit combinatorial auctions," in *Proc. 7th Int. Joint Conf. Auton. Agents Multiagent Syst.*, Richland, SC, USA, 2008, pp. 1601–1604.

[13] A. Giovannucci, M. Vinyals, J. A. Rodríguez-Aguilar, and J. Cerquides, "Computationally-efficient winner determination for mixed multi-unit combinatorial auctions," in *Proc. 7th Int. Joint Conf. Auton. Agents Multiagent Syst.*, Estoril, Portugal, 2008, pp. 1071–1078.

[14] A. Giovannucci, J. Cerquides, and J. A. Rodríguez-Aguilar, "Composing supply chains through multiunit combinatorial reverse auctions with transformability relationships among goods," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 40, no. 4, pp. 767–778, Jul. 2010.

[15] C. Crick and P. Pfeffer, "Loopy belief propagation as a basis for communication in sensor networks," in *Proc. 19th Conf. Uncertainty Artif. Intell. (UAI)*, Acapulco, Mexico, 2003, pp. 159–166.

[16] T. Voice, R. Stranders, A. Rogers, and N. Jennings, "A hybrid continuous max-sum algorithm for decentralised coordination," in *Proc. 19th Eur. Conf. Artif. Intell. (ECAI)*, Amsterdam, The Netherlands, 2010, pp. 61–66.

[17] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, vol. 1, 1st ed. San Francisco, CA, USA: Morgan Kaufmann, 1988.

[18] D. MacKay, *Information Theory, Inference, and Learning Algorithms*, vol. 1, 1st ed. Cambridge, U.K.: Cambridge Univ. Press, 2003.

[19] Y. Weiss, "Correctness of local probability propagation in graphical models with loops," *Neural Comput.*, vol. 12, pp. 1–41, Jan. 2000.

[20] M. Vinyals, J. Cerquides, A. Farinelli, and J. A. Rodríguez-Aguilar, "Worst-case bounds on the quality of max-product fixed-points," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, pp. 2325–2333.

[21] R. McEliece, D. MacKay, and C. Jung-Fu, "Turbo decoding as an instance of Pearl's 'belief propagation' algorithm," *IEEE J. Sel. Areas Commun.*, vol. 16, no. 2, pp. 140–152, Feb. 1998.

[22] B. Frey and D. MacKay, "A revolution: Belief propagation in graphs with cycles," in *Proc. Neural Inf. Process. Syst. Conf.*, Denver, CO, USA, 1998, pp. 479–485.

[23] P. Felzenszwalb and D. Huttenlocher, "Efficient belief propagation for early vision," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2004, pp. 261–268.

[24] A. Farinelli, A. Rogers, A. Petcu, and N. Jennings, "Decentralized coordination of low-power embedded devices using the max-sum algorithm," in *Proc. 7th Int. Joint Conf. Auton. Agents Multiagent Syst.*, Richland, SC, USA, 2008, pp. 639–646.

[25] T. Penya-Alba, M. Vinyals, J. Cerquides, and J. A. Rodriguez-Aguilar, "A scalable message passing algorithm for supply chain formation," in *Proc. 26th Conf. Artif. Intell.*, Toronto, ON, Canada, 2012.

[26] T. Penya-Alba, J. Cerquides, J. A. Rodriguez-Aguilar, and M. Vinyals, "CHAINME: Fast decentralized finding of better supply chains," in *Proc. Int. Joint Conf. Auton. Agents Multiagent Syst.*, Richland, SC, USA, 2013, pp. 1317–1318.

[27] M. Winsper and M. Chli, "Using the max-sum algorithm for supply chain formation in dynamic multi-unit environments," in *Proc. 11th Int. Joint Conf. Auton. Agents Multiagent Syst.*, Valencia, Spain, 2012, pp. 1285–1286.

[28] E. W. Dijkstra and C. S. Scholten, "Termination detection for diffusing computations," *Inf. Process. Lett.*, vol. 11, no. 1, pp. 1–4, 1980.

[29] M. Sharma, S. Iyengar, and N. Mandyam, "An efficient distributed depth-first-search algorithm," *Inf. Process. Lett.*, vol. 32, no. 4, pp. 183–186, 1989.

[30] W.-C. Yeh and M. El Khadiri, "A new universal generating function method for solving the single (d, τ)-quick-path problem in multistate flow networks," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 42, no. 6, pp. 1476–1484, Nov. 2012.

Maria Chli received the M.Eng. and the Ph.D. degrees in computing from Imperial College London, London, U.K., in 2001 and 2005, respectively.

She is currently a Senior Lecturer with the Department of Computer Science, School of Engineering and Applied Sciences, Aston University, Birmingham, U.K. Her current research interests include multiagent systems and modeling of complex systems using agent-based methodologies, to help in understanding the dynamics and in analyzing the emergent properties, and ultimately to aid in decision-making and policy formation, multiagent systems to design and evaluate mechanisms that make real-life complex systems more efficient.

Michael Winsper received the B.Sc. and Ph.D. degrees in computing from Aston University, Birmingham, U.K., in 2008 and 2012, respectively.

He is currently a Postdoctoral Researcher with the Supply Chain Improvement Group, University of Derby, Derby, U.K. His current research interests include multiagent systems, supply chain management, social simulation, and sensor networks.